On the Effects of Diversity on Intrusion Tolerance

Alysson Neves Bessani
Rafael Obelheiro
Paulo Sousa
Ilir Gashi

DI–FCUL TR–08–30

December 2008

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

Technical reports are available at `http://www.di.fc.ul.pt/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

On the Effects of Diversity on Intrusion Tolerance

Alysson Neves Bessani[1] Rafael Obelheiro[2] Paulo Sousa[1] Ilir Gashi[3]
[1]LaSIGE, University of Lisbon, Portugal
[2]Universidade do Estado de Santa Catarina, Brazil
[3]Centre for Software Reliability, City University London, United Kingdom

Abstract

The security gains of intrusion-tolerant systems are directly dependent on the assumption that system components fail independently of one another. The coverage of this assumption in a real-world deployment depends on how diversity is employed, using, for example, diverse off-the-shelf components. In this paper we detail a study we have done with vulnerability data, reported in the period 1999 to 2007, which we extracted from the NIST National Vulnerability Database. We provide empirical analysis of the data collected as well as exploratory analyses of the potential gains in security from employing diverse operating systems. The modelling approaches presented are of practical significance to system designers wishing to employ diversity with off-the-shelf components since often the vulnerability reports are the only direct security evidence available to them.

Keywords: Diversity, Intrusion Tolerance, Byzantine Fault Tolerance, Security.

1 Introduction

Intrusion-tolerant systems, which are systems that are able to keep functioning correctly even if some of their parts are compromised, usually rely on Byzantine fault-tolerant protocols to coordinate system components. Such protocols guarantee correct behavior in spite of arbitrary faults provided that a minority (usually less than one third) of components are faulty [18]. To respect this condition, system components need to fail independently of each other (at least to a certain extent). However, when security is considered, the possibility of simultaneous attacks against several components cannot be dismissed. If multiple components exhibit the same vulnerabilities, they can be compromised by a single attack, which defeats the whole purpose of building a replication-based intrusion-tolerant system in the first place. To avoid this, *diversity* can be employed: each component uses different software to perform the same functions, with the expectation that the differences will reduce the occurrence of common vulnerabilities.

Nearly all software systems built today rely on off-the-shelf (OTS) components, such as operating systems and database management systems. This is mostly due to the sheer complexity of such components, coupled with benefits such as the perceived lower costs from their use (some of the components may be open-source and/or freely available), faster deployment and the multitude of available options. Most OTS software, however, have not been designed with security as their top priority, which means that they all have their share of security flaws that can be exploited. At times, supposedly secure systems are compromised not due to vulnerabilities in application software but in a more surreptitious manner, by compromising a critical component in their software infrastructure. On the other hand, given the ready availability of OTS software, leveraging OTS components to implement diversity is less complex and more cost-effective than actually developing variants of software. One of the prime examples is the operating system: realistically, people will deploy an OTS operating system rather than build their own. Given the variety of operating systems available and the critical role played by the OS in any system, diversity at the OS level can be a reasonable way of providing good security against common vulnerabilities at little extra cost. The question we address in this paper is how to measure the commonality of vulnerabilities in OTS software, which can then be used to evaluate the benefits provided by a diversified system deployment.

We have collected vulnerability data from the National Vulnerability Database (NVD) [14] reported in the period 1999 to 2007 for 7 different operating systems. We focus our study on operating systems for several reasons: they offer a good opportunity for diversity, many intrusions exploit OS vulnerabilities, and the number of OS-related vulnerability reports in the NVD is sufficiently large to give meaningful results. Each vulnerability report in the NVD database contains (amongst other things) information about which operating systems the vulnerability affects. We collected this data and checked how many vulnerabilities affect more than one operating system. We found this number to be relatively low for most pairs of operating systems in our study. These results seem to suggest that security gains may be achieved if

diverse operating systems are employed in replicated systems. However they are not definitive evidence. The main problem is that the available reports concern vulnerabilities and not how many intrusions or exploits occurred for each vulnerability; this makes their use in security evaluation more difficult. Complete intrusions and exploit rates would be much more useful as statistical evidence, but they are not widely available. To most practitioners the only direct security evidence available for these products often are the vulnerability reports.

It is the lack of detailed intrusion and exploit data and the lack of known approaches that can utilize existing vulnerability reports of OTS components in security evaluation that has motivated the research detailed in this paper. One of the present authors has also looked at a similar problem with the lack of failure data for reliability predictions [6]. The question we attempt to answer is "how can we incorporate existing evidence for off-the-shelf software to evaluate the similarity of these software when it comes to their security properties?" To this end we have studied two approaches which utilize vulnerability reports for obtaining measures of the security benefits of using diverse software, and applied these approaches to popular operating systems. The two approaches are:

1. A "proportions" approach where the observed reliability of a single software is scaled by a factor to derive the expected gains in security from running alongside it another diverse software. This is based on the approach first reported in [6].

2. A "vulnerability similarity" approach where a single measure is obtained for pairs of replicas based on data about vulnerabilities in the software used by the replicas. The lower the score obtained for a pair, the less similar the software packages that make up the pair are in terms of their common vulnerabilities.

We illustrate the use of the two approaches with the data collected from NVD.

The rest of the paper is organized as follows: section 2 provides a description of the empirical study with the data from the national vulnerability database; section 3 shows the details of the two approaches used to derive estimates of the security benefits from using diverse operating systems: both approaches use the data from the NVD study as evidence in calculating the estimates; section 4 provides a discussion of the approaches, the results obtained and the assumptions made; section 5 reviews related work on modelling diversity and empirical studies with faults and vulnerabilities and finally section 6 provides conclusions and provisions for further work.

2 Description of the data used for the study

In order to get initial estimates of the potential benefits of diversity we have analyzed data from the National Vulnerability Database (NVD) [14]. NVD is a US-government sponsored database of reported, confirmed and analyzed vulnerabilities. NVD aggregates vulnerability reports from several important security companies, forums, advisory groups and organizations, being thus one of the most reliable and complete vulnerability databases on the web. All data is made available on the web as XML files containing all reported vulnerabilities on a given period, called data feeds. As of November 2008, it contained more than $33,300$ vulnerabilities encompassing the period from 1999 to 2008.

Each NVD data feed contains a list of reported vulnerabilities sorted by its date of publication on a given period. For each vulnerability, called entry in NVD, we have several interesting data:

- An unique name for the entry, in the format CVE-*YEAR-NUMBER*;
- The list of products (with version numbers) affected by the vulnerability;
- The date of the vulnerability publication;
- The CVSS score of the vulnerability (0–10) and the associated sub-scores (base, exploit and impact) [13]. This score defines how important is this vulnerability;
- The severity of the score (low, medium or high), as assessed by NVD analysts;
- The type of the vulnerability (access, input, design, exception, race, etc.);
- The security attribute(s) that is(are) lost when this vulnerability is exploited on a system (*avail*, *int*, *conf* and *sec_prot*);
- From where this vulnerability can be exploited (*local*, *local_network*, *network*, *user_init*).

Despite the large amount of information about each vulnerability available on NVD, for the purposes of this paper, we are interested only in the name, publication date and list of affected products of each vulnerability. In our study we have collected vulnerabilities reported for the kernels of 7 operating systems. More than 2000 vulnerabilities, reported between 1999 and 2007, were collected for these operating systems.

Table 1 shows the raw data collected from NVD: *vulns(A)* shows the total number of vulnerabilities collected for a given OS A, *vulns(B)* counts vulnerabilities for a given OS B, whereas *vulns(AB)* is the count of vulnerabilities that affect both systems A and B (and hence is a subset of counts A and B).

Further analysis of this data will be presented in the next section.

3 Exploratory analyses of the benefits of diversity

In this section we will describe two approaches to analyze the potential gains in security from using diverse software,

OS Pairs (A-B)	*vulns*(*A*)	*vulns*(*B*)	*vulns*(*AB*)
FreeBSD - Linux	229	437	11
FreeBSD - MacOS		405	9
FreeBSD - NetBSD		121	39
FreeBSD - OpenBSD		131	45
FreeBSD - Solaris		411	18
FreeBSD - Windows2k		347	3
Linux - MacOS	437	405	0
Linux - NetBSD		121	7
Linux - OpenBSD		131	4
Linux - Solaris		411	5
Linux - Windows2k		347	3
MacOS - NetBSD	405	121	5
MacOS - OpenBSD		131	6
MacOS - Solaris		411	4
MacOS - Windows2k		347	0
NetBSD - OpenBSD	121	131	33
NetBSD - Solaris		411	16
NetBSD - Windows2k		347	2
OpenBSD - Solaris	131	411	11
OpenBSD - Windows2k		347	2
Solaris - Windows2k	411	347	3

Table 1. Operating Systems' reported vulnerabilities between 1999 and 2007.

and show how these approaches can be applied using vulnerability data on operating systems. Both of the approaches attempt to get away from the need to quantify usage time and attack rates on a system, and derive measures based on the vulnerability counts alone using data from the NVD. This is because data on attack as well as exploit rates for a given vulnerability are difficult to obtain. We discuss these issues further in the next subsection before we describe the approaches used to analyze the data and obtain estimates on benefits of diversity.

3.1 Estimating Intrusions

According to the Attack-Vulnerability-Intrusion-error-failure (AVI) model [16], which is the security extension of the classical dependability fault-error-failure model [4], a security failure lies at the end of a causal chain of an attack exploiting a dormant vulnerability which leads to an intrusion in the system, which in turn may cause an error in the system state and finally a failure of the system to deliver its service according to its specification and/or security policy. Let $P(A_V, r)$ be the probability of an attack A_V that exploits some vulnerability V on replica r and $P(V, r)$ be the probability of this replica being vulnerable to V. Considering the AVI model, and for mathematical convenience excluding the time dimension of usage time from the equation, we may then define the probability of an intrusion on a system replica r with the following equation:

$$P(I, r) = P(A_V, r)P(V, r) \quad (1)$$

Unfortunately, it is not possible to obtain this probability directly from the data available in the NVD. The probability of an attack A_V on r ($P(A_V, r)$) cannot be readily estimated from publicly available data; even the best information sources available, such as security incident reports from CERT [7] and attack data collected from honeypots [9], are very limited in scope, and the attack profile observed against systems running on one network cannot be easily translated to another system running on a different network.

It is the absence of attack and intrusion data that has motivated us to define and adapt the two methods we outline in the following sections. We have tried to answer the question "how can we incorporate existing evidence for operating systems to evaluate the possible gains in security that can be achieved through the use of diversity". We try to answer this question utilizing the most reliable publicly available security evidence that we could find about the operating systems: the vulnerability data, such as that available in NVD.

3.2 Proportions approach

3.2.1 Description of the proportions approach

Bishop and colleagues [6] describe an approach which attempts to get away from the need to quantify actual usage time between failures. The model is used to get initial estimates of the potential gains in reliability from switching from a software A to a diverse system AB which runs two diverse redundant software A and B using fault counts alone. In what follows we give a brief description of this model and its underlying assumptions and how it can be applied in a security context by using vulnerability counts as evidence. The "proportions" approach, defined in [6] to model the reliability of a 1-out-of-2 system[1], uses:

- The counts of faults which are available from the fault logs of each product. These are the dependability equivalent of the vulnerability counts for security which we have collected from the NVD. From these counts we can then calculate the proportion of vulnerabilities in product A that are also found to exist in product B, β_{AB}, from the ratio of common to non-common vulnerabilities in the vulnerability history of A. Similarly we can also calculate β_{BA} for product B vulnerabilities that are also found to exist in A. Since we have looked at a joint vulnerability database, namely NVD, rather than individual OS vulnerability databases then the AB and BA counts in our study are the same. However the β_{AB} and β_{BA} can still be different depending on the total number of vulnerabilities for each OS.

[1] A system in which the system works correctly as long as one of the two diverse redundant components that make up the system continues to provide the correct service.

- The pfd (probability of failure on demand) of the products A and/or B; the equivalent in our study would be the probability of an intrusion on a system, which we already mentioned that it is difficult to obtain from the publicity available data, but estimates of these may exist for a particular system deployment based on actual intrusions in operation for that system.

This approach has the following underlying assumptions:

- Common vulnerabilities are drawn from the same intrusion rate distribution as non common vulnerabilities, i.e. a constant proportion of vulnerabilities in each intrusion rate band are common to A and B.
- The intrusion rate distributions for A and B are the same.

Given these assumptions (see the discussion in [6] for the validity of these assumptions), we can estimate the expected common mode intrusion rate as:

$$E(\lambda_{AB}) = \beta_{AB} E(\lambda_A) or \tag{2}$$

$$E(\lambda_{BA}) = \beta_{BA} E(\lambda_B) \tag{3}$$

Where $E(\lambda_{AB})$ and $E(\lambda_{BA})$ both represent common mode intrusion rate estimates that should be, in principle, equivalent. In what follows we will describe in more detail how these two expressions were obtained in [6], replacing the term fault with vulnerability where necessary.

3.2.2 The underlying theory of the proportions approach

In [6] it was stated that the *fault density*, $h(\phi)$, represents the number of faults within a given failure rate interval that remain in an OS. If we use the same expression for vulnerability density then the OS vulnerability count and intrusion rate are given by:

$$N = \int_0^\infty h(\phi)\, d\phi \qquad \text{and} \qquad \lambda = \int_0^\infty h(\phi)\phi\, d\phi,$$

respectively.

We assume the *vulnerability density functions* of the A, B and AB fault classes are:

$$\begin{aligned} h(\phi)_A &= N_A p(\phi) \\ h(\phi)_B &= N_A p(\phi) \\ h(\phi)_{AB} &= N_{AB} p(\phi) \end{aligned}$$

where:

- N_A is the total number of vulnerabilities in Product A;
- N_B is the total number of vulnerabilities in Product B;
- N_{AB} are vulnerabilities common to Products A and B;
- $p(\phi)$ is the probability distribution[2] of intrusion rate ϕ for a vulnerability in the product (assumed to be the same for A, B and AB vulnerabilities).

Note that N_A and N_B here are the total number of vulnerabilities in each product.

Under these assumptions, the expected number of those vulnerabilities n_{A,τ_A} observed in product A during usage until time τ_A is:

$$E(n_{A,\tau_A}) = N_A \left(1 - \int_0^\infty p(\phi) e^{-\phi\tau_A}\, d\phi\right) \tag{4}$$

The expected value of the number of vulnerabilities n_{AB,τ_A} observed in product A during usage until time τ_A that are also common to product B is:

$$E(n_{AB,\tau_A}) = N_{AB} \left(1 - \int_0^\infty p(\phi) e^{-\phi\tau_A}\, d\phi\right) \tag{5}$$

It can be seen that the assumption of a common intrusion rate distribution means that the bracketed term (the probability a vulnerability is found after time τ_A) is identical for $E(n_A)$ and $E(n_{AB})$ and will cancel out if we take the ratios. So knowledge of the actual usage time τ_A and the intrusion rate distribution $p(\lambda)$ is not required.

So we can estimate β_{AB} from the vulnerability sequence observed in product A up to τ_A, where some vulnerabilities in the sequence are labeled as being common to B (from a knowledge of the B product vulnerabilities). Given the observed values, n_{A,τ_A} and n_{AB,τ_A}:

$$\beta_{AB} = N_{AB}/N_A \approx n_{AB,\tau_A}/n_{A,\tau_A} \tag{6}$$

Similarly, we can also estimate β_{BA} from the vulnerability sequence observed in product B up to τ_B

$$\beta_{BA} = N_{AB}/N_B \approx n_{AB,\tau_B}/n_{B,\tau_B} \tag{7}$$

These β values need not necessarily be identical as one product could contain more vulnerabilities than another.

3.2.3 Empirical derivations of β

Table 2 shows empirical derivations of β for the OS pairs in our study. The table also contains 90% upper confidence bounds on the estimates. The confidence bound is computed using

$$\Pr(\beta < p | n, x) = \sum_{r=0}^{x} C(n,r) p^r (1-p)^{n-r} \tag{8}$$

where x is the number of common vulnerabilities in a sequence of n vulnerabilities.

[2] We use λ for the intrusion rate of an entire *program* (i.e. Product A, Product B or 1-out-of-2 AB intrusion rate), and we use ϕ for the intrusion rate of a randomly chosen *vulnerability*.

OS pair	β_{AB}	90% bound	β_{BA}	90% bound
FreeBSD - Linux	0.048	0.072	0.025	0.038
FreeBSD - MacOS	0.039	0.061	0.022	0.035
FreeBSD - NetBSD	0.170	0.206	0.322	0.383
FreeBSD - OpenBSD	0.197	0.234	0.344	0.402
FreeBSD - Solaris	0.079	0.107	0.044	0.060
FreeBSD - Windows2k	0.013	0.029	0.009	0.019
Linux - MacOS	0.000	0.005	0.000	0.006
Linux - NetBSD	0.012	0.023	0.041	0.075
Linux - OpenBSD	0.009	0.018	0.031	0.060
Linux - Solaris	0.010	0.020	0.010	0.019
Linux - Windows2k	0.000	0.006	0.000	0.007
MacOS - NetBSD	0.012	0.023	0.041	0.075
MacOS - OpenBSD	0.015	0.026	0.046	0.079
MacOS - Solaris	0.010	0.020	0.010	0.019
MacOS - Windows2k	0.000	0.006	0.000	0.007
NetBSD - OpenBSD	0.273	0.331	0.252	0.307
NetBSD - Solaris	0.132	0.181	0.039	0.054
NetBSD - Windows2k	0.017	0.043	0.006	0.015
OpenBSD - Solaris	0.084	0.124	0.027	0.040
OpenBSD - Windows2k	0.015	0.040	0.006	0.015
Solaris - Windows2k	0.007	0.016	0.009	0.019

Table 2. Estimates of β for each OS pair

The β values vary considerably for the different OS pairs. This is because of the genuine difference between the operating systems in our study: different flavours of BSD operating systems have larger β values since these operating systems started from the same common base, hence it would be expected that they have larger common vulnerabilities. Win2k is different from the other OSs and hence has lower β values for most pairs. From equations (6) and (7), it can be seen that the β_{AB}, β_{BA} values need not be identical as they depend on the number of residual vulnerabilities, N_A and N_B, which can vary with the quality of the security development process. However many of the β_{AB}, β_{BA} values for the OS product pairs seem to be similar given the inherent sampling errors.

Taking the data set as a whole, the results suggest that for most diverse OS product pairs, β values of 0.1 (and possibly lower) are possible. This means that using a 1-out-of-2 OS configuration reduces the common intrusion rate 10 fold or more compared with a single OS product.

3.3 VSS approach

The second approach we outline in this paper tries to obtain a single *vulnerability similarity score* (VSS) for the different operating system pairs used in our study. This approach also uses the vulnerability counts alone to derive the measures. It does this by combining the β values obtained in the previous section. The lower the VSS score obtained for a pair the better candidates the operating systems that make up the pair would be to be used in a diverse configuration, as they would be expected to have a lower number of similar vulnerabilities. Hence, with the model in section 3.2 one can calculate which operating system B would have the least number of vulnerabilities in common with an operating system A which is already being run in a system deployment (and for which detailed intrusion and attack rates may be available). The model to be outlined in this section is useful in cases where the decision on which operating system to use in a new system deployment has not been made yet and the system designer is interested in getting a single similarity measure of the different operating systems based on the vulnerability counts, so that they can decide which pair of OSs to choose.

3.3.1 Model and assumptions

We consider an intrusion-tolerant (or Byzantine fault-tolerant) system $\Pi = \{r_0, r_1, ..., r_{n-1}\}$ with n replicas. The replicas of the system can be classified according to a set of diversity axes [15], including location, operating system, hardware, etc. For the purpose of this paper we are only interested in the software-related axes, i.e., application (or the service being replicated), middleware (mainly a replication library) and operating system.

This way, each replica i can be seen as a set of software packages $r_i = \{s_0, s_1, ..., s_{k-1}\}$ with k software packages that can have vulnerabilities. The set of vulnerabilities of a software package s_j is denoted by $vuln(s_j)$.

The VSS methodology is based on the following assumptions:

- The number of reported vulnerabilities is an approximation of the number of vulnerabilities of a given software system. This assumption holds for systems that have not been released too recently, since reliability models shows that the number of vulnerabilities of a given system tends to stabilize after some time [2].
- The number of common vulnerabilities that appear on different software systems is an approximation of how similar the two systems are in terms of fault dependence (i.e., how often an attack that compromises one system can compromise the other).

3.3.2 Vulnerability Similarity Score

As we already discussed in section 3.1, it is not feasible to calculate the probability of an intrusion on a given replica without making many strong assumptions about the system (even assuming that attacks will eventually happen, i.e., $P(I,r) \approx P(V,r)$). However, given two replicas r_i and r_j of a system, it is possible to calculate what is the probability that a vulnerability affecting r_i also affects r_j by looking at the software packages used by the replicas and their reported vulnerabilities on NVD. This probability is given by the number of common vulnerabilities in r_i and r_j divided by the number of vulnerabilities in r_j. This reflects the proportion of common vulnerabilities with respect to vulnerabilities affecting only r_j, and is expressed by the following equation:

$$P((V,r_i)|(V,r_j)) = \frac{\#\bigcup_{s \in r_i \cap r_j} vuln(s)}{\#\bigcup_{s \in r_j} vuln(s)} \quad (9)$$

Notice that in general $P((V,r_i)|(V,r_j)) \neq P((V,r_j)|(V,r_i))$, since the amount of reported vulnerabilities for a given replica (the denominator in Eq. (9)) can be different. Also, if we consider a single software package per replica, $P((V,r_i)|(V,r_j))$ is equivalent (conceptually and numerically) to $\beta_{r_j r_i}$.

We are interested in having a single value that measures how similar two replicas are in terms of vulnerabilities. So, we define the *vulnerability similarity score* (VSS) of two replicas r_i and r_j ($vss(r_i,r_j)$) as

$$vss(r_i,r_j) = P((V,r_i)|(V,r_j) \vee (V,r_j)|(V,r_i)) \quad (10)$$

Considering $P_{i,j} = P((V,r_i)|(V,r_j))$ and $P_{j,i} = P((V,r_j)|(V,r_i))$, we can calculate the VSS of replicas i and j with the following:

$$vss(r_i,r_j) = P_{i,j} + P_{j,i} - P_{i,j}P_{j,i} \quad (11)$$

The $vss(r_i,r_j)$ of two replicas can be interpreted as "the probability that an adversary discovers a vulnerability of r_i and successfully exploits it on r_i and r_j *or* discovers a vulnerability on r_j and successfully exploits it on r_j and r_i".

The VSS metric takes into account only what we can currently measure, and all the few assumptions made are conservative. In addition, it has several good properties that make it an attractive measure of the similarity between different replicas of an intrusion-tolerant system:

1. It is relatively simple to calculate;
2. It takes into account several software packages running on a replica;
3. Replicas with the same software packages have VSS = 1, while completely different replicas (with no known common vulnerabilities) have VSS = 0;
4. If the same software package (e.g., application, BFT library, JVM) runs on more than one of the evaluated replicas, it contributes to increase the VSS proportionally to the amount of vulnerabilities it has.

In the next section we apply this similarity to several operating systems to get a feeling about the level of vulnerability dependencies that may be expected in an intrusion tolerant system if diverse OSs are used on its replicas.

3.3.3 Using the VSS to Measure the Similarity of Diverse BFT Systems

Scenario. Let us assume that there is a BFT/intrusion-tolerant system in which all replicas use the same application software (with no reported vulnerabilities), the same BFT replication library (also with no reported vulnerabilities) and different operating systems. Given the high costs of using N-version programming and the difficulties in integrating diverse OTS application software in general, this is a rather cost-effective solution for practical BFT systems[3] [5].

VSS of operating systems. Considering the pragmatic BFT service above, one question that we want to answer is: if we deploy the various replicas with different operating systems, how much more secure will the system be? One way to answer this question is to calculate the VSS of each pair of operating systems, using Eq. (11) with the data from Table 1. Figure 1 depicts these values.

There are at least two interesting things that we can observe on Figure 1: *(1.)* the different BSD flavors are approximately 50% similar; *(2.)* Windows 2K has little similarity (at most 2%) with other operating systems. *(1.)* can be explained by the fact that all BSDs came originally from the same codebase. *(2.)* can be explained by the fact that, contrary to the other six OSs considered, Windows 2K is not an UNIX-like operating system, so one should expect a somewhat different organization of its codebase.

Now, let us consider the four most diverse operating systems for the classical case of $3f+1$ replicas with $f=1$. If we analyze Figure 1, it is easy to see that the most interesting choice would be FreeBSD, Linux, Solaris and Windows 2K. OpenBSD and MacOS could also be considered instead of FreeBSD and Windows 2K, but we want to choose OSs as popular as possible, and avoid non-server market OSs (like MacOS). Considering this choice of OSs, does some vulnerability affect more than one replica? By inspecting Figure 1 the similarity between any two of these four OSs (i.e., the probability of a given attack to affect two replicas of our pragmatic BFT system) is at most 12%.

The evolution of the number of reported vulnerabilities and the VSS. To better evaluate the effectiveness of VSS as an approximation of how diverse a system can be, we analyzed the cumulative VSS calculated for several OSs from the year 1999 up to the year 2007 and verified how it compares with the number of accumulated vulnerabilities of these OSs on these years. Figures 2 and 3 plots the cumulative number of vulnerabilities and VSS of the four most-diverse OSs and the four least-diverse OSs, respectively.

These figures clearly show that, as the number of reported vulnerabilities increase, the VSS of replicas with these operating systems decrease. This means that the VSS metric appears to be a conservative approximation of how often a vulnerability in one system could be exploited on another, i.e., despite the fact that we measure VSS based on historical data, our results show that, in general, it is expected that the similarity of pairs of replicas will decrease in the future. In this case, by

[3]The resources spent on producing different versions of a given application would be put on the test and verification task of a single high quality version that would be deployed in all replicas.

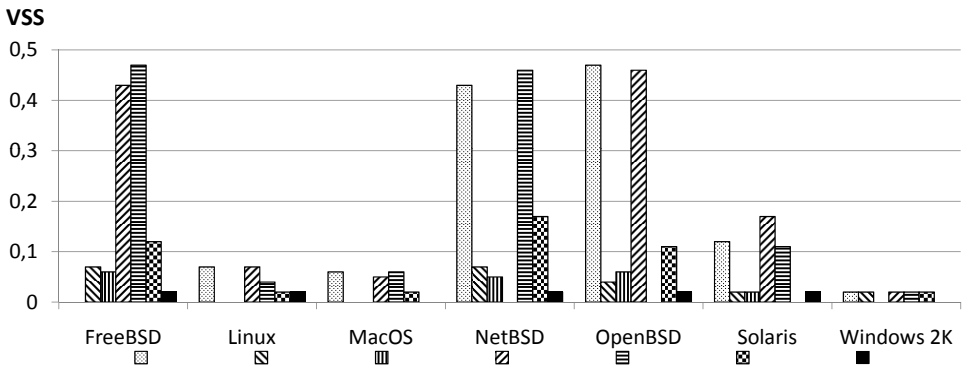
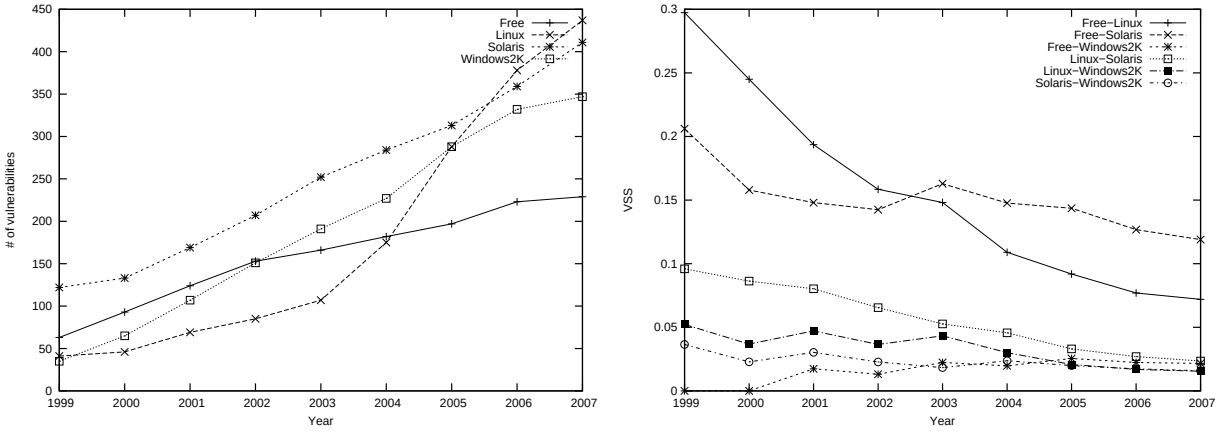
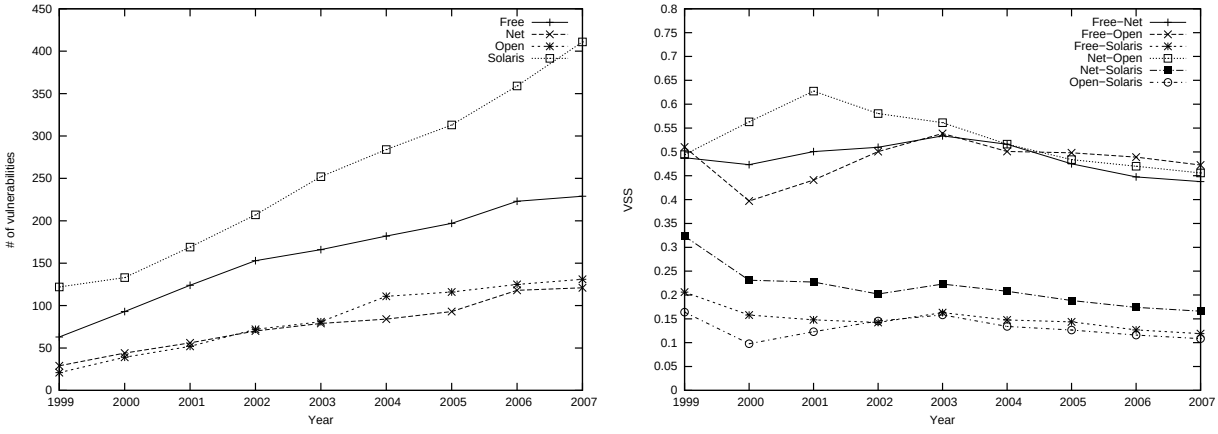


Figure 1. Vulnerability similarity scores of pairs of replicas with diverse operating systems.



(a) Cumulative number of vulnerabilities. (b) Cumulative VSS.

Figure 2. Vulnerabilities and VSS of most diverse OSs from 1999 to 2007.



(a) Cumulative number of vulnerabilities. (b) Cumulative VSS.

Figure 3. Vulnerabilities and VSS of least diverse OSs from 1999 to 2007.

"conservative" we mean that the VSS used to make decisions today will be lower in the future, so that a good decision now will not turn out to be a bad one later on.

The history/observed experiment. In order to evaluate the effectiveness of our approach, we did a simple experiment using the NVD data set. The rationale of our experiment was the following: what if, at some point in the past, we evaluated the NVD database using the VSS metric and based on our observations we defined a runtime environment for a seven replica IT system. The interesting question here is, how beneficial would have been this strategy, i.e., would it have been effective in suggesting a set of OSs with a low similarity? To answer this question we divided the NVD data in three periods and computed 2/3 as *history* measured to define our runtime environment (1999–2004) and 1/3 as the VSS *observed* on the three years that the system was to be used (2005–2007).

Figures 4 and 5 show the VSS of the history (Figure 4) and observed (Figure 5) data sets. From these figures it can be seen that most of the VSS values decreased from the history data set to the observed data set. Only 2 pairs out of 21 possible (Windows2K/FreeBSD and Windows2K/OpenBSD) had an increase on their VSS. This means that only for these pairs of systems, the probability of someone finding a vulnerability in one of them and exploiting it on the other increased. However, this increase was very modest: Windows2K/FreeBSD goes from 2% to 3% and Windows2K/OpenBSD goes from 1% to 6%.

To complement the results presented in Figures 4 and 5, it is worth to report also the number of common vulnerabilities observed on each pair of operating systems both between 1999 and 2004 and between 2005 and 2007. Table 3 shows these numbers.

The pair of numbers on the diagonal of the table shows the total number of vulnerabilities of an operating system on the two data sets (history/observed). The table shows that the OS pairs that had an increase on their VSS values (bold values) had in fact the same number or even less common vulnerabilities on the observed period. However, the overall number of vulnerabilities of the operating systems on these pairs is much lower on the observed data set than on history data set, so it increases the probability of some vulnerability of these systems being common.

Effectiveness of Diversity. Our simple experiment with the NVD data set illustrates how diversity can be used to boost the intrusion tolerance of a BFT system. Consider a BFT replicated service that need $3f + 1$ replicas to tolerate f faults, and consider the case in which $f = 1$ (four replicas). If all four replicas of this system use OpenBSD, the operating system with less vulnerabilities reported on the observed period (assuming that we were able to predict how many vulnerabilities the operating systems would present in the next three years at the beginning of 2005), we would have observed *at least* 20 vulnerabilities that, when exploited, could bring the whole system down (20 is the number of OpenBSD vulnerabilities in the observed period). On the other hand, if we had chosen to employ diverse operating systems using FreeBSD, Linux, Solaris and Windows2K, we would have observed *at least* three vulnerabilities (maximum VSS is 12% and the total number of vulnerabilities is 20) that would be able to bring down multiple replicas, from an universe of 556 reported vulnerabilities (i.e., less than 0.5% of the vulnerabilities affect more than one of these four OSs). In both cases, the number of vulnerabilities is actually a lower bound, since there might be vulnerabilities in the systems discussed that were not publicly reported.

3.3.4 Summary of the Results

From the results presented in this section several points can be highlighted:

1. When considering the whole universe of vulnerabilities reported on NVD (from 1999–2008), one can see that pairs of operating systems from the BSD family (FreeBSD, NetBSD and OpenBSD) have the greatest VSS value of 43–47%; Windows2K, on the other hand, is very different from all other operating systems (a VSS of only 2%);

2. Based on the reported vulnerabilities of NVD we can recommend the following operating systems for a $3f + 1$ BFT system that tolerates one fault: Windows2K, Linux, FreeBSD and Solaris. For $f > 1$ our study shows that it becomes difficult to avoid common vulnerabilities (at least at the OS level): the addition of the other three systems to our replica set brings VSS scores up to 46%;

3. Our history/observed experiment was an attempt to travel back in time and evaluate the effectiveness of our approach if we had applied it at the beginning of 2005 and evaluated the results at the beginning of 2008. The experiment shows that the approach would be successful: only two out of 21 OS pairs had an increase on their VSS, and this increase was very small (1% and 5%, in terms of VSS absolute values).

4 Discussion

Developing adequate metrics for computer security has long been a challenge. There is relevant work done in the area of reliability [12], but the main approaches require a model of the demand profile of the system. When we move to the security domain, leveraging the existing work is hard because it is very difficult to estimate the attack profile of a system with a reasonable degree of confidence, as pointed out in Sec. 3.1. Both approaches introduced in this paper provide ways of estimating the security offered by diversified systems without resorting to models of attacker behavior (under the assumptions mentioned in Sec. 3.1). This, combined with the lack of other metrics for diversity, highlights the relevance of our contributions, even if they are first steps that can be improved.

	FreeBSD	Linux	MacOS	NetBSD	OpenBSD	Solaris	Windows2K
FreeBSD	182/47	10	7	34	40	17	**2**
Linux	1	262/175	0	6	3	5	3
MacOS	2	0	292/113	4	6	4	0
NetBSD	5	1	1	84/37	29	14	2
OpenBSD	5	1	0	4	111/20	11	**1**
Solaris	1	0	0	2	0	284/127	3
Windows2K	**1**	0	0	0	**1**	0	227/120

Table 3. Number of common faults of pairs of replicas with diverse operating systems considering the periods of 1999–2004 ("history" – top/right) and 2005–2007 ("observed" – down/left). The diagonal shows the total number of vulnerabilities on the two periods (history/observed).

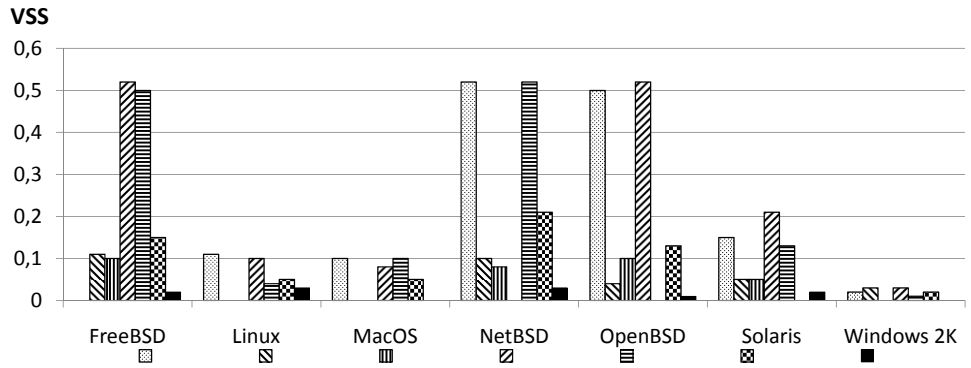


Figure 4. Vulnerability similarity scores of pairs of replicas with diverse operating systems (1999–2004).

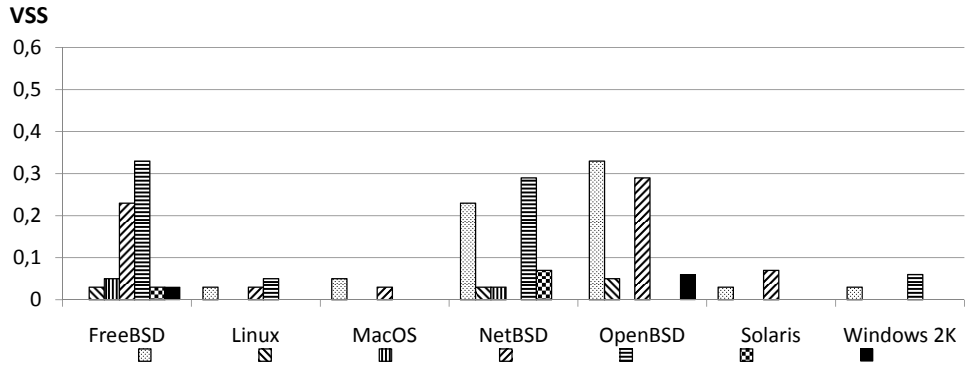


Figure 5. Vulnerability similarity scores of pairs of replicas with diverse operating systems (2005–2007).

One of our foremost concerns with the proposed metrics is how faithfully they represent reality. On closer inspection both the β values and the VSS obtained reveal that these values agree with our prior expectations. For instance, all BSD systems are derivations of a single source tree (this is even more true of NetBSD and OpenBSD, since the latter was forked from the former); this is reflected by high β and VSS values for them, which would indicate that these systems are very similar and hence the gains in added security from employing these systems in a diverse setup would be minimal. The frequent cross-pollination between open source OSs results in greater commonality. Of the seven OSs studied, Windows 2000 has the least commonality with others, not only in terms of source code but also in terms of internal structure

and interface with userland; as consequence, it has lower β and VSS values compared to the other systems. Although our results agree with common sense expectations, they are based on hard data, and not on anecdotal evidence. When comparing systems whose origins cannot be readily traced, our methods can provide useful information about their similarities.

It is also important to understand how our results are influenced by the available data. First, we have restricted our analysis to kernel vulnerabilities, but an operating system comprises the kernel and a set of userland tools. The problem here is that, especially for open source OSs, it is difficult to determine exactly which tools should be considered as part of the OS and which should be left out. Since we were aiming at a general analysis that could demonstrate the usefulness of our approaches, we decided to limit ourselves to the kernel. However, it is not only possible but quite feasible to estimate the similarity of more complete systems for specific scenarios where the set of tools to be used is known beforehand.

Another issue with the data is that the NVD database does not contain all existing vulnerabilities, but only those that have been reported. For systems with large user bases and that have been released for some time, the NVD data should be reasonably complete (see, e.g., [2]), and therefore the similarity scores should be sufficiently accurate. For less popular or more recent systems, there might be larger numbers of undiscovered vulnerabilities, and the accuracy of similarity scores should suffer. However, it must be emphasized that the NVD database is one of the most complete sources of vulnerability information available (in fact, we are not aware of a better source that is freely available).

5 Related Work

Littlewood and colleagues [11] survey a number of issues in software diversity modeling, presenting models that have been developed for assessing the reliability of systems that adopt diversity. The models discussed aim to provide a measure of the reliability of a system as a function of the demands presented to the system and how these demands influence the correctness of the behavior of the system; these parameters are, for the most part, expressed as probability distributions. They show that, although diversity does not provide complete failure independence (since design faults are correlated to some extent), it is an effective means of increasing overall system reliability. They also discuss a number of caveats regarding software diversity modeling.

An experimental study of the benefits of adopting diversity of SQL database servers is presented in [8]. The authors analyzed bug reports for four database servers and verified which products were affected by each bug reported (the focus of their study is on overall dependability, not specifically on security). They found few cases of a single bug affecting more than one server, and that there were no coincident failures in more than two of the servers. Their conclusion is that diversity of off-the-shelf database servers is an effective means of improving system reliability.

A comparison of the robustness of four different operating systems is presented in [10]. This study was based on fault injection: combinations of valid and invalid parameters were supplied to often-used systems calls, and the effects of this on reliability (e.g., system crash, process hang/crash, wrong error returned, etc.) were observed. The authors found out some commonalities among the systems studied; however, from the available data it is impossible to conclude whether there were specific bugs that affected more than one system (the paper only shows how many failures were observed for each system call in several degrees of severity).

Some vulnerability discovery models, which attempt to forecast the amount of vulnerabilities found in software, have been proposed [3, 17, 1]. Alhazmi and Malayia [2] investigate how well these models fit with vulnerability data from the NVD, and conclude that the vulnerability discovery process follows the same S-shaped curve of "traditional" software reliability growth models [12], which measure all defects found in a system (not only those that affect security). While their study cross-validates our idea of using the NVD as a source of vulnerability data, they are more concerned in modeling how many vulnerabilities are found in specific software over its lifetime, and our focus is on assessing the degree of independence between different operating systems.

6 Conclusions

In this paper we have presented two approaches for estimating the gains in security from using diverse operating systems. We use vulnerability counts alone to derive the estimates. These approaches are based on some strong assumptions about the operational profile and intrusion/attack rate distributions which may not hold in real operation. Ideally we would like to have detailed information about intrusion and attack counts and usage time. However detailed intrusion and attack data are rarely available even to the software vendors themselves. Also due to the various non-restrictive license agreements of the open-source products, an operating system may be downloaded from a multitude of sources and then installed in many different instances, which makes estimation of the usage time of an operating system very difficult. Faced with these difficult problems of data availability, it was necessary to make these strong modeling assumptions in order to make an initial estimate of the potential security benefits of diversity with operating systems.

The estimates obtained from the two approaches confirm some of our initial expectations: that operating systems from the same code base (BSD flavors) are predicted to have higher common vulnerabilities and hence would bring less gains in security if used together; but Windows 2000, has very few common vulnerabilities with other operating systems and would benefit from running it alongside another operating

system in a diverse configuration. Overall in most cases, given the assumptions made in the paper, we can expect a ten-fold improvement in security from running diverse operating systems in a replicated system.

Further research is needed to validate the theory presented in this paper. This research includes:

- Methods for obtaining more accurate proxies for usage time of the operating systems.
- Empirical investigations of the predictive performance of the two approaches for actual operating system pairs.
- Empirical investigations of the consistency of the β and VSS estimates in successive releases of the same product pair.
- Applying the methods to other types of off-the-shelf components (such as diverse web-servers and application servers).

Acknowledgments

This work was partially supported by the EC through project IST-2004-27513 (CRUTIAL) and NoE IST-4-026764-NOE (RESIST), and by the FCT, through the CMU-Portugal project and the Multiannual Funding Programme.

References

[1] O. H. Alhazmi and Y. K. Malayia. Quantitative vulnerability assessment of systems software. In *Proceedings of the Annual Reliability and Maintainability Symposium*, pages 615–620, Jan. 2005.

[2] O. H. Alhazmi and Y. K. Malayia. Application of vulnerability discovery models to major operating systems. *IEEE Transactions on Reliability*, 57(1):14–22, Mar. 2008.

[3] R. J. Anderson. Security in open versus closed systems—the dance of Boltzmann, Coase and Moore. In *Conference on Open Source Software: Economics, Law and Policy*, Toulouse, France, 2002.

[4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan./Mar. 2004.

[5] A. N. Bessani, E. P. Alchieri, M. Correia, and J. S. Fraga. DepSpace: a Byzantine fault-tolerant coordination service. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Systems Conference*, Apr. 2008.

[6] P. Bishop, I. Gashi, B. Littlewood, and D. Wright. Reliability growth modelling of a 1-out-of-2 system: Research with diverse off-the-shelf SQL database servers. In *Proc. IEEE International Symposium on Software Reliability Engineering (ISSRE'07)*, pages 49–58, Trollhåttan, Sweden, Nov. 2007.

[7] CERT/CC. `http://www.cert.org/`, 2008.

[8] I. Gashi, P. Popov, and L. Strigini. Fault tolerance via diversity for off-the-shelf products: A study with SQL database servers. *IEEE Transactions on Dependable and Secure Computing*, 4(4):280–294, Oct./Dec. 2007.

[9] The Honeynet Project. `http://www.honeynet.org/`, 2008.

[10] P. Koopman, J. Sung, C. Dingman, D. Siewiorek, and T. Marz. Comparing operating systems using robustness benchmarks. In *Proceedings of the 16th Symposium on Reliable Distributed Systems (SRDS)*, pages 72–79, Durham, NC, USA, Oct. 1997.

[11] B. Littlewood, P. Popov, and L. Strigini. Modeling software design diversity: A review. *ACM Computing Surveys*, 33(2):177–208, 2001.

[12] M. R. Lyu, editor. *Handbook of Software Reliability Engineering*. McGraw-Hill, 1995.

[13] P. Mell, K. Scarfone, and S. Romanosky. Common vulnerability scoring system. *IEEE Security & Privacy*, 4(6):85–89, Nov./Dec. 2006.

[14] National Vulnerability Database. `http://nvd.nist.gov/`, 2008.

[15] R. R. Obelheiro, A. N. Bessani, L. C. Lung, and M. Correia. How practical are intrusion-tolerant distributed systems? DI/FCUL TR 06–15, Department of Informatics, University of Lisbon, September 2006. Available from `http://www.di.fc.ul.pt/tech-reports/06-15.pdf`.

[16] D. Powell and R. Stroud, editors. *Conceptual Model and Architecture of MAFTIA*. Deliverable D21. MAFTIA Project, Jan. 2003. `http://www.laas.research.ec.org/maftia/deliverables/D21.pdf`.

[17] E. Rescorla. Is finding security holes a good idea? *IEEE Security & Privacy*, 3(1):14–19, Jan./Feb. 2005.

[18] P. Verissimo, N. F. Neves, and M. P. Correia. Intrusion-tolerant architectures: Concepts and design. In *Architecting Dependable Systems*, volume 2677 of *LNCS*. 2003.